Neural Networks.
Lab 4: Radial Basis Functions (RBF) Networks. Regression and prediction

---

1. Defining RBF networks.

There are four main functions used to design RBF networks: newrbe, newrb, newgrnn, newpnn. In all cases when the neural network is created its parameters (centers corresponding to the hidden units, widths associated to Gaussian activation functions, weights of connections to the output units) are also set. The four variants are different with respect to the way they establish the values of the centers and weights. In all cases the parameter width is explicitly set by the user.

a) newrbe(inputs, desiredOutputs, width): creates a RBF neural network which has as many hidden units as examples are in the training set. The centers (parameters associated to the connections between the input and the hidden layers) are set to the input values in the training set (inputs). The weights corresponding to the output units are computed using the same approach as in the case of one-layer linear networks (as in the case of function newlind).
b) newrb(inputs, desiredOutputs, goal, width): creates a RBF neural network in an incremental way: at each step there is added a new hidden unit having as center an input vector from the training set (it is selected the input vector which generates the largest error). New hidden units are added until the goal specified by the user is reached or until or inputs in the training set have been added. The weights corresponding to the output units are computed as in the case of newrbe.
c) newgrnn(inputs, desiredOutputs, width): creates a network which realizes an exact interpolation of the data in the training set. The network has as many hidden units as examples are in the training set. The centers are the inputs from the training set and the weights of connections between the hidden and the output layers are the corresponding desired outputs from the training set. The name of the function derives from "Generalized Regression Neural Network".
d) newpnn(inputs, desiredOutputs, width): creates networks used for classification problems (the desiredOutputs have just the component corresponding to the desired class equal to 1; all othe components are 0). The parameters are set as in the case of newgrnn and the values generated by the hidden layer are normalized (this means that they can be interpreted as probabilities)

Disregarding the network type the simulation is realized by using the function sim(network, inputs). In the case of probabilistic neural networks in order to obtain the index of the output class one have to apply the function vec2ind to the output produced by the network.

2. Applications

2.1. Nonlinear regression

We revisit the regression problem analyzed in Lab 3. Let us consider a set of bidimensional data represented as points in plane (the coordinates are specified by using the function ginput). Find a function which approximates the data (by minimizing the sum of squared distances between the points and the graph of the function). The function regressionRBF described below allows to get the coordinates of points to be included in the training set, creates the network in an incremental way and plot the regression function.

```
function [in,d]=regressionRBF(goal, width)
clf
axis([0 1 0 1])
hold on
in = [];
d = [];
n = 0;
b = 1;
while b == 1
[xi,yi,b] = ginput(1);
plot(xi,yi,'r*');
n = n+1;
in(1,n) = xi;
d(1,n) = yi;
end
inf=min(in); sup=max(in);
ret=newrb(in,d,goal,width);
x=inf:(sup-inf)/100.:sup;
y=sim(ret,x);
plot(in,d,'b*',x,y,'r-');
end
```

Exercise:

1. Analyze the influence of the value of parameters width on the quality of regression. In order to use the same set of data for all tests first network will be created by using the function regressionRBF while the other tests will use the function regressionRBFin (which receives the training set as first parameters). For instance the first call can be:

   [input,d]=regressionRBF(0.05, 1);

   and a subsequent call (using the same training set but different values of the parameters) can be:

   regressionRBFin(input,d, 0.05, 0.01);

2.2. Prediction

Let us revisit the problem of predicting the next value in a time series (see Lab3). Let us consider a sequence of data (a time series): $x_1,x_2,\ldots,x_n$ which can be interpreted as values recorded at successive moments of time. The goal is to predict the value corresponding to moment (n+1). The main idea is to suppose that a current value x_i depends on N previous values: x_{i-1}, $x_{i-2},\ldots,x_{i-N}$. Based on this hypothesis we can design a neural network which is trained to extract the association between any subsequence of L and the next value in the series.

Therefore, the neural network will have N input units, a given number of hidden units and 1 output unit. The training set will have n-N pairs of (input data, correct output): $\{((x_1,x_2,\ldots,x_N),x_{N+1}),((x_2,x_3,\ldots,x_{N+1}),x_{N+2}),\ldots,((x_{n-N},x_{n-N+1},\ldots,x_{n-1}),x_n)\}$. The problem can be solved by a RBF network as is specified in function predictionRBF.

The data should be first read by using the function csvread:

data=csvread('date.dat')';

```
function [in,d]=predictionRBF(data,inputUnits,goal,width)
% data: row array containing the time series
% inputUnits: it depends on the number of previous values which
influence
%              the current value
% goal: the maximal accepted error on the training set
% width: parameter controlling the extension of tha gaussians used as
%         activation functions for the hidden layer
L=size(data,2);
in=zeros(inputUnits,L-inputUnits);
d=zeros(1,L-inputUnits);
for i=1:L-inputUnits
    in(:,i)=data(1,i:i+inputUnits-1);
    d(i)=data(1,i+inputUnits);
end
ret=newrb(in,d,goal,width);
x=1:L-inputUnits;
y=sim(ret,in);
inTest=zeros(1,inputUnits);
inTest=date(1,L-inputUnits+1:L)';
rezTest=sim(ret,inTest);
disp('Predicted value:'); disp(rezTest);
plot(x,d,'b-',x,y,'r-',L+1,rezTest,'k*');
end
```

Call example: predictionRBF(data,5,0.1,10);

Exercises.

1. Analyze the influence of the number of input units on the ability of the network to make prediction (Hint: try the following values: 2, 5, 10, 15)
2. Analyze the influence of the parameter width on the prediction ability of the network (Hint: try the following values: 0.01, 0.1, 1, 10, 50, 100)
3. Compare the results obtained by using a BackPropagation network (see Lab 3) with the results obtained by using predictionRBF.